说点题外话04｜面向对象的原则适用于RESTful API吗？

2021-07-27 徐昊

《如何落地业务建模》 课程介绍 >

讲述：徐昊
时长 04:31 大小 10.38M

你好，我是徐昊。今天我们再来专门说点题外话。

前面几期题外话都比较偏向于提供一种不同的角度，主要是因为你们也并没有针对课程的内容，提出什么特别的问题需要我来具体回答。那么作为我们在进入新约前的最后一篇题外话，我想聊一聊关于 RESTful API 的问题。

我记得有位同学在留言区问了这样一个问题：过长的 URI 是否破坏了迪米特法则（Law of Demeter）。这里我们就要搞清楚，什么是迪米特法则呢？

迪米特法则又叫最小可知法则，指的是**在面向对象设计中，实体应尽可能少地与其他实体发生交互**。为了说明什么是“少的交互”，我们还特别归纳了一组可以认为不违反迪米特法则，并且可以直接调用的对象：

当前对象自己（this，self）；

以参数形式传入的对象，比如函数的形参（parameter）；

当前对象内实例变量引用的对象（instance variable）；

如果实例变量是集合，那么集合中的对象也可以访问（collection，aggregration）；

由当前对象创建的对象（variable declaration in function）。

那么这些场景适用于 RESTful API 调用的场景吗？显然并不太适用。因为在 RESTful API 的场景中，实体只有客户端和 API 提供者，而 API 提供者的内在结构都被 API 层屏蔽了。所以无论怎么调用，都不会出现对于 API 提供者内部结构的依赖。

如果我们把迪米特法则扩展一下，推广到概念层面，暴露了内在的逻辑就算。那么恰恰是 RESTful API，能帮助我们继续遵循迪米特法则。

比如通过超媒体明确地表示资源之间的关联，而不是依靠客户端去拼凑 URI。如果客户端可以拼凑出 URI，则表明客户端对于 API 提供者的内在逻辑存在依赖。而通过 HATEOAS，把所有关联的链接直接提供，就避免了暴露内在的逻辑。

再比如，API 的设计要按照 HTTP 语义约定，而不是客户端与 API 供应者之间的“私约”（private protocol）。PUT 只能修改已经存在的资源，而不能构建新的资源；POST 创建资源成功，需要返回 201，并在 HEAD 中给出新构建资源的 URI；GET 默认都是可以缓存的，无法缓存的查询（而不是通过 URI 遍历信息），需要用 POST 访问等等。这些都是希望通过公约，将客户端中对于 API 供应者的了解降到最低。

所以迪米特法则本身的想法是不错的，但是场景改变了，我们就要重新思考它在新的环境中是如何被应用的。而如果我们真的在乎迪米特法则，那么在 RESTful API 的场景下，关注点就不会放在 URI 有多长上，而应放在客户端与服务器间的知识依赖到底有多少上。

另外一个对于 RESTful API 的质疑在于，通过 HATEOAS 完全以分布式超媒体的方式构成 API，那么客户端看起来越来越像浏览器，而不是针对 RESTful API 的客户端了。

正如我们在第 10 讲中讲到的，RESTful 架构风格**是对互联网架构的反思**。那么互联网架构的核心在于开放性和扩展性，因而 **RESTful 架构风格的核心也是开放性和扩展性**。

因为开放，使得 RESTful API 的供应者不会对客户端作出任何假设。就好像互联网服务器并不会假设它的客户端只有浏览器一样，wget、telnet 等等都是可能的客户端；而因为扩展性，RESTful API 只会为客户端提供最基本的功能，大量的计算被分布到了客户端侧进行。

这种架构的假设是不同于企业应用的客户端与服务器架构的，在企业应用架构的语境中，客户端与服务器有更多的耦合。服务端更多地是为客户端提供服务，而不是保持自己开放和稳定。

那么为什么 RESTful API 最终还是成为“行业主流”了呢（虽然真的会，和真的用的人并不多）？

因为从大趋势上来说，将企业内的能力（而不仅仅是后台）构造成开放 API，并围绕着开放 API，形成企业内生态是大势。在这个大势之下，RESTful API、MicroService、企业内生态、能力平台、中台形成了一条清晰的企业架构现代化之路。仅仅服务于某些（或者某个）前台的后台服务，终将会淡出历史的舞台。

因而在这种历史转折的节点，我们更应该清晰地理解不同想法之间的差异，哪怕它们要解决的问题与现行方法是相似的，但是对于这是什么问题，我们要怎么想，还是带来了完全不同的角度。

思考题

请问其他面向对象原则与最佳实践，在 RESTful 架构下有何种体现？

欢迎把你的思考和想法分享在留言区，我会和你交流。我们新约部分再见！

分享给需要的人，Ta订阅后你可得 20 元现金奖励

赞 2 提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇
下一篇 12 | 云时代的挑战（上）：弹性边界还是业务边界？

精选留言 (2)

写留言

阿鸡
2021-08-01

想请问下SOA能使用面向对象的原则吗？ 还是只能在服务内部面向对象，在服务边界还是传统的DO加XxService的形态

展开

邹溪源
2021-07-27

接口无处不在，而良好的设计则更显得非常重要。基于restful的架构风格设计接口，相比传统的http风格设计的接口，更易于引入面向对象的程序设计。这种设计也成为实践单一职责原则的练兵场。
而有了接口，也使得依赖倒置成为一种基本操作。

展开

作者回复: 面向对象不面向对象单说 比较容易得到稳定的api